UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/550,851	09/07/2007	Ying'an Deng	42P21031	6410

45209 7590 09/28/2010

INTEL/BSTZ
BLAKELY SOKOLOFF TAYLOR & ZAFMAN LLP
1279 OAKMEAD PARKWAY
SUNNYVALE, CA 94085-4040

EXAMINER	
SWIFT, CHARLES M	

ART UNIT	PAPER NUMBER
2191	

MAIL DATE	DELIVERY MODE
09/28/2010	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/550,851	DENG ET AL.	
	Examiner	Art Unit	
	Charles Swift	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *07 September 2007*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1 - 25* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1 - 25* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *9/22/2005* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *11/3/2006, 9/14/2009*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This is the initial Office Action based on the application filed on 9/7/2007.

2. Claims 1 - 25 are pending.

Priority

3. Applicant has claimed priority to **PCT/CN04/01586 filed on 12/31/2004**, the priority is confirmed and the effective filing date of this application is **12/31/2004**.

Claim Rejections - 35 USC § 102

4. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

5. Claims **1, 2, 4 – 6, 11, 13 – 17 and 20** are rejected under 35 U.S.C. 102(b) as being anticipated by **Wygodny et al (USPAT 6202199, hereinafter Wygodny)**.

As per claim 1, Wygodny discloses: A method, comprising:

- executing a program code on a first computer system; (Wygodny col 4, line 54 – 57, "`"Tracing," or "to trace," refers generally to the process of using a monitoring program to monitor and record information about the execution of the client while the`

client is running." Note a client program is a computer program that is being debugged)

- generating debug information upon the occurrence of an error during execution of the program code; (Wygodny col 6, line 1 – 9, "Once attached, the agent 104 extracts trace information, such as execution paths, subroutine calls, and variable usage, from the client 102. Again, the TCI file 120 contains instructions to the client-side trace library 125 regarding the trace data to collect. The trace data collected by the client-side trace library 125 is written to the trace buffer 105. On command from the user 110 (such as when a bug manifests itself), the agent 104 copies the contents of the trace buffer 105 to a trace log file 122.")

- and transmitting the debug information to a second computer system via a network adaptor. (Wygodny col 6, line 11 – 12, "The user 110 sends the trace log file 122 back to the developer 112." And col 3, line 26 – 44, "In one embodiment, the software system provides a remote mode that enables the client program to be traced at a remote site, such as by the customer at a remote customer site, and then analyzed at the developer site. When the remote mode is used, the developer sends the TCI file for

the particular client to a remote user site together with a small executable file called the tracing "agent." The agent is adapted to be used at the remote user site as a stand-alone tracing component that enables a remote customer, who does not have access to the source code of the client, to generate a trace file that represents execution of the client application at the remote site. The trace file is then sent to the developer site (such as by email), and is analyzed by the software developer using the analyzer. The remote mode thus enables the software developer to analyze how the client program is operating at the remote site, without the need to visit the remote site, and without exposing to the customer the source code or other confidential details of the client program.")

As per claim 2, Wygodny discloses:

- The method of claim 1, wherein generating debug information is performed by executing a function call in the program code to a network print driver. (Wygodny col 6, line 1 – 9, "Once attached, the agent 104 extracts trace information, such as execution paths, subroutine calls, and variable usage, from the client 102. Again, the TCI file

120 contains instructions to the client-side trace library 125 regarding the trace data to collect. The trace data collected by the client-side trace library 125 is written to the trace buffer 105. On command from the user 110 (such as when a bug manifests itself), the agent 104 copies the contents of the trace buffer 105 to a trace log file 122. In some cases, the log data is written to a file automatically, such as when the client terminates. The user 110 sends the trace log file 122 back to the developer 112. As shown in FIG. 1C, the developer 112 then uses the analyzer 106 to view the information contained in the trace log file 122." Note that Wygodny discloses a method for developer to access to trace information of a client computer remotely, thus internet is needed for the trace information to be transmitted back to the developer, the network print driver is just a network protocol to transfer file from one computer to another according to applicant's specification.)

As per claim 4, Wygodny discloses:

- The method of claim 1, further comprising: building a debug information node from the debug information. (Wygodny col 9, line 39 – 53, "The analyzer 106 comprises a User Interface module that reads trace data,

either from the trace buffer 105 (during on-line mode tracing) or from the trace log file 122 (e.g. after remote tracing) and displays the data in a format, such as a trace tree, that shows the sequence of traced events that have occurred during execution of the client 102. Much of the trace data comprises assembly addresses. With reference to FIG. 1C, the analyzer 106 uses the debug information 121 to translate the traced assembly addresses to comprehensive strings that are meaningful to the developer. In order to save memory and gain performance, this translation to strings is preferably done only for the portion of the trace data which is displayed at any given time, not the whole database of trace data. Thus, for example, in formatting a screen display in the user interface, only the trace data needed for the display in the user interface at any given time is read from the log file 122. This allows the analyzer 106 to display data from a trace log file 122 with more than a million trace records." Note a debug node is just debug information from a particular client.)

As per claim 5, Wygodny discloses:

- The method of claim 4, wherein the debug information node includes data selected from the group consisting of: priority, time stamp, host ID, metadata, separator, and debug information. (Wygodny col 9, line 39 – 53, "The analyzer 106 comprises a User Interface module that reads trace data, either from the trace buffer 105 (during on-line mode tracing) or from the trace log file 122 (e.g. after remote tracing) and displays the data in a format, such as a trace tree, that shows the sequence of traced events that have occurred during execution of the client 102. Much of the trace data comprises assembly addresses. With reference to FIG. 1C, the analyzer 106 uses the debug information 121 to translate the traced assembly addresses to comprehensive strings that are meaningful to the developer. In order to save memory and gain performance, this translation to strings is preferably done only for the portion of the trace data which is displayed at any given time, not the whole database of trace data. Thus, for example, in formatting a screen display in the user interface, only the trace data needed for the display in the user interface at any given time is read from the log file 122. This allows the analyzer 106 to display data from a trace log file 122 with more than a million trace records.")

As per claim 6, Wygodny discloses:

- The method of claim 5, wherein the metadata includes data selected from the group consisting of: module name, sub-module name, priority, file name, and line number. (Wygodny col 8, line 21 – 33, "The debug information 121 is preferably created by a compiler when the client is compiled. Using the debug information 121 the analyzer translates function names and source lines to addresses when creating the TCI file 120. Conversely, the analyzer 106 uses the debug information 121 to translate addresses in the trace data back into function names and source lines when formatting a display for the user interface. One skilled in the art will recognize that other build information may be used as well, including, for example, information in a linker map file and the Type Library information available in a Microsoft OLE-compliant executable.')

As per claim 11, Wygodny discloses: A method, comprising:

- receiving debug information from a computer program at a filter and node builder; (Wygodny col 6, line 1 – 9, "Once attached, the agent 104 extracts

trace information, such as execution paths, subroutine calls, and variable usage, from the client 102. Again, the TCI file 120 contains instructions to the client-side trace library 125 regarding the trace data to collect. The trace data collected by the client-side trace library 125 is written to the trace buffer 105. On command from the user 110 (such as when a bug manifests itself), the agent 104 copies the contents of the trace buffer 105 to a trace log file 122." And col 8, line 40 – 49, "The analyzer 106 allows the developer 112 to open multiple trace tree windows and define a different filter (trace control instructions) for each of window. When reading a trace record, each window filter is preferably examined separately to see if the record should be displayed. The filters from the various windows are combined in order to create the TCI file 120, which is read by the client-side trace library 125. In other words, the multiple windows with different filters are handled by the User Interface, and the client-side trace library 125 reads from a single TCI file 120.")

- building a node of debug information using configurable parameters from a configuration module; (Wygodny col 9, line 39 – 53, "The analyzer 106 comprises a User Interface module that reads trace data,

either from the trace buffer 105 (during on-line mode tracing) or from the trace log file 122 (e.g. after remote tracing) and displays the data in a format, such as a trace tree, that shows the sequence of traced events that have occurred during execution of the client 102. Much of the trace data comprises assembly addresses. With reference to FIG. 1C, the analyzer 106 uses the debug information 121 to translate the traced assembly addresses to comprehensive strings that are meaningful to the developer. In order to save memory and gain performance, this translation to strings is preferably done only for the portion of the trace data which is displayed at any given time, not the whole database of trace data. Thus, for example, in formatting a screen display in the user interface, only the trace data needed for the display in the user interface at any given time is read from the log file 122. This allows the analyzer 106 to display data from a trace log file 122 with more than a million trace records." Note a debug node is just debug information from a particular client.)

- transmitting the node through a network adaptor using a scheduler. (Wygodny col 6, line 11 – 12, "The user 110 sends the trace log file 122 back to the developer 112." And col 3, line 26 – 44, "In one

embodiment, the software system provides a remote mode that enables the client program to be traced at a remote site, such as by the customer at a remote customer site, and then analyzed at the developer site. When the remote mode is used, the developer sends the TCI file for the particular client to a remote user site together with a small executable file called the tracing "agent." The agent is adapted to be used at the remote user site as a stand-alone tracing component that enables a remote customer, who does not have access to the source code of the client, to generate a trace file that represents execution of the client application at the remote site. The trace file is then sent to the developer site (such as by email), and is analyzed by the software developer using the analyzer. The remote mode thus enables the software developer to analyze how the client program is operating at the remote site, without the need to visit the remote site, and without exposing to the customer the source code or other confidential details of the client program.")

As per claim 13, Wygodny discloses:

- The method of claim 11, further comprising: filtering debug information at the filter and node builder using the configurable parameters from the configuration module. (Wygodny col 8, line 40 – 49, "The analyzer 106 allows the developer 112 to open multiple trace tree windows and define a different filter (trace control instructions) for each of window. When reading a trace record, each window filter is preferably examined separately to see if the record should be displayed. The filters from the various windows are combined in order to create the TCI file 120, which is read by the client-side trace library 125. In other words, the multiple windows with different filters are handled by the User Interface, and the client-side trace library 125 reads from a single TCI file 120.")

As per claim 14, Wygodny discloses:

- The method of claim 13, wherein the configurable parameters are selected from the group consisting of: priority, time stamp, host ID, metadata, separator, debug information, module name, sub-module name, priority, file name, line number, project name, and serial number. (Wygodny col 8, line 21 – 33, "The debug information 121 is preferably created by a compiler when the client is compiled. Using the debug information 121 the

analyzer translates function names and source lines to addresses when creating the TCI file 120. Conversely, the analyzer 106 uses the debug information 121 to translate addresses in the trace data back into function names and source lines when formatting a display for the user interface. One skilled in the art will recognize that other build information may be used as well, including, for example, information in a linker map file and the Type Library information available in a Microsoft OLE-compliant executable.')

As per claim 15, Wygodny discloses: An article of manufacture, comprising: a machine-readable medium on which a plurality of instructions are stored, which when executed perform operations comprising: executing a program code stored in a first computer system;

- building a debug information node upon the occurrence of an error during execution of the program code; (Wygodny col 6, line 1 – 9, "Once attached, the agent 104 extracts trace information, such as execution paths, subroutine calls, and variable usage, from the client 102. Again, the TCI file 120 contains instructions to the client-side trace library 125 regarding the trace data to collect. The trace data collected by the client-

side trace library 125 is written to the trace buffer 105. On command from the user 110 (such as when a bug manifests itself), the agent 104 copies the contents of the trace buffer 105 to a trace log file 122." And col 8, line 40 – 49, "The analyzer 106 allows the developer 112 to open multiple trace tree windows and define a different filter (trace control instructions) for each of window. When reading a trace record, each window filter is preferably examined separately to see if the record should be displayed. The filters from the various windows are combined in order to create the TCI file 120, which is read by the client-side trace library 125. In other words, the multiple windows with different filters are handled by the User Interface, and the client-side trace library 125 reads from a single TCI file 120." and col 9, line 39 – 53, "The analyzer 106 comprises a User Interface module that reads trace data, either from the trace buffer 105 (during on-line mode tracing) or from the trace log file 122 (e.g. after remote tracing) and displays the data in a format, such as a trace tree, that shows the sequence of traced events that have occurred during execution of the client 102. Much of the trace data comprises assembly addresses. With reference to FIG. 1C, the analyzer 106 uses the debug information 121 to

translate the traced assembly addresses to comprehensive strings that are meaningful to the developer. In order to save memory and gain performance, this translation to strings is preferably done only for the portion of the trace data which is displayed at any given time, not the whole database of trace data. Thus, for example, in formatting a screen display in the user interface, only the trace data needed for the display in the user interface at any given time is read from the log file 122. This allows the analyzer 106 to display data from a trace log file 122 with more than a million trace records." Note a debug node is just debug information from a particular client.)

- and invoking a network print driver to transmit the debug information node to a second computer system through a network adaptor. (Wygodny col 6, line 1 – 9, "Once attached, the agent 104 extracts trace information, such as execution paths, subroutine calls, and variable usage, from the client 102. Again, the TCI file 120 contains instructions to the client-side trace library 125 regarding the trace data to collect. The trace data collected by the client-side trace library 125 is written to the trace buffer 105. On command from the user 110 (such as when a bug manifests itself), the agent 104 copies the

contents of the trace buffer 105 to a trace log file 122. In some cases, the log data is written to a file automatically, such as when the client terminates. The user 110 sends the trace log file 122 back to the developer 112. As shown in FIG. 1C, the developer 112 then uses the analyzer 106 to view the information contained in the trace log file 122." Note that Wygodny discloses a method for developer to access to trace information of a client computer remotely, thus internet is needed for the trace information to be transmitted back to the developer, the network print driver is just a network protocol to transfer file from one computer to another according to applicant's specification.)

As per claim 16, Wygodny discloses:

- The article of manufacture of claim 15, wherein the debug information node includes data selected from the group consisting of: priority, time stamp, host ID, metadata, separator, and debug information. (Wygodny col 8, line 21 – 33, "The debug information 121 is preferably created by a compiler when the client is compiled. Using the debug information 121 the analyzer translates function names and source lines to addresses when creating the TCI file 120. Conversely, the analyzer 106 uses the debug information 121 to

translate addresses in the trace data back into function names and source lines when formatting a display for the user interface. One skilled in the art will recognize that other build information may be used as well, including, for example, information in a linker map file and the Type Library information available in a Microsoft OLE-compliant executable.')

As per claim 17, Wygodny discloses:

- The article of manufacture of claim 16, wherein the metadata includes data selected from the group consisting of: module name, sub-module name, priority, file name, and line number. (Wygodny col 8, line 21 – 33, "The debug information 121 is preferably created by a compiler when the client is compiled. Using the debug information 121 the analyzer translates function names and source lines to addresses when creating the TCI file 120. Conversely, the analyzer 106 uses the debug information 121 to translate addresses in the trace data back into function names and source lines when formatting a display for the user interface. One skilled in the art will recognize that other build information may be used as well, including, for

example, information in a linker map file and the Type Library information available in a Microsoft OLE-compliant executable.')

As per claim 20, Wygodny discloses: A computer system, comprising: a processor; a network adaptor operatively coupled to the processor; at least one flash device operatively couple to the processor on which firmware instructions are stored; and at least one storage device on which computer program code is stored, which when executed by the processor performs operations comprising:

- receiving debug information from a computer program upon the occurrence of an error during execution of the program code; (Wygodny col 6, line 1 – 9, "Once attached, the agent 104 extracts trace information, such as execution paths, subroutine calls, and variable usage, from the client 102. Again, the TCI file 120 contains instructions to the client-side trace library 125 regarding the trace data to collect. The trace data collected by the client-side trace library 125 is written to the trace buffer 105. On command from the user 110 (such as when a bug manifests itself), the agent 104 copies the contents of the trace buffer 105 to a trace log file 122.")

- applying configuration parameters to the debug information to create a debug information node; (Wygodny col 9, line 39 – 53, "The analyzer 106 comprises a User Interface module that reads trace data, either from the trace buffer 105 (during on-line mode tracing) or from the trace log file 122 (e.g. after remote tracing) and displays the data in a format, such as a trace tree, that shows the sequence of traced events that have occurred during execution of the client 102. Much of the trace data comprises assembly addresses. With reference to FIG. 1C, the analyzer 106 uses the debug information 121 to translate the traced assembly addresses to comprehensive strings that are meaningful to the developer. In order to save memory and gain performance, this translation to strings is preferably done only for the portion of the trace data which is displayed at any given time, not the whole database of trace data. Thus, for example, in formatting a screen display in the user interface, only the trace data needed for the display in the user interface at any given time is read from the log file 122. This allows the analyzer 106 to display data from a trace log file 122 with more than a million trace records." Note a debug node is just debug information represents a particular client.)

- and transmitting the debug information node via the network adaptor to a remote computer. (Wygodny col 6, line 1 – 9, "Once attached, the agent 104 extracts trace information, such as execution paths, subroutine calls, and variable usage, from the client 102. Again, the TCI file 120 contains instructions to the client-side trace library 125 regarding the trace data to collect. The trace data collected by the client-side trace library 125 is written to the trace buffer 105. On command from the user 110 (such as when a bug manifests itself), the agent 104 copies the contents of the trace buffer 105 to a trace log file 122. In some cases, the log data is written to a file automatically, such as when the client terminates. The user 110 sends the trace log file 122 back to the developer 112. As shown in FIG. 1C, the developer 112 then uses the analyzer 106 to view the information contained in the trace log file 122.")

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claim **3** is rejected under 35 U.S.C. 103(a) as being unpatentable over **Wygodny**, in view of **Willems (USPAT 6983452)**.

As per claim 3, Wygodny further discloses:

- The method of claim 2, further comprising: transmitting the debug information to the network print driver; (Wygodny col 3, line 26 – 44, "In one embodiment, the software system provides a remote mode that enables the client program to be traced at a remote site, such as by the customer at a remote customer site, and then analyzed at the developer site. When the remote mode is used, the developer sends the TCI file for the particular client to a remote user site together with a small executable file called the tracing "agent." The agent is adapted to be used at the remote user site as a stand-alone tracing component that enables a remote customer, who does not have access to the source code of the client, to generate a trace file that represents execution of the client application at the remote site. The trace file is then sent to the developer site (such as by email), and is analyzed by the software developer using the analyzer. The remote mode thus enables the software developer to analyze how the client program is operating at the remote site, without the need to visit the remote site, and without exposing to the customer the

source code or other confidential details of the client program.")

Wygodny did not discloses:

- halting execution of the program code during execution of the function call to the network print driver; and resuming execution of the program code after transmitting the debug information to the network print driver.

However, Willems teaches:

- halting execution of the program code during execution of the function call to the network print driver; and resuming execution of the program code after transmitting the debug information to the network print driver. (Willems col 5, line 8 – 20, "The collection driver of the symbolic kernel debugger is operating system specific. It is capable of taking snapshots of the target machine system state according to an input record list provided by the user interface command plug-in. Coherency of each snapshot is enforced through use of an operating system call to suspend execution of all processes and threads other than the collection driver itself for the time it takes to capture the snapshot. After the snapshot is collected, the suspended processes and threads are restarted so that operation can continue. This

frequently permits capture of snapshots without visible disruption of system operation")

It would have been obvious for one of ordinary skill in the art at the time of invention to incorporate the teaching of Willems into that of Wygodny in order to halting execution of the program during error report and resume execution after the error report is finished. This approach would ensure the debugger or tracer capture an accurate snapshot the status of the program and thus gives the remote debugger an accurate view on the exact condition which caused the error to occur, this allows the remote debugger a better understanding of the problem.

8. Claims **7 and 18** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Wygodny**, in view of **Brouwer et al (USPAT 6279124, hereinafter Brouwer)**.

As per claim 7, Wygodny did not disclose:

- The method of claim 6, wherein the separator includes data selected from the group consisting of: project name and serial number.

However, Brouwer teaches:

- wherein the separator includes data selected from the group consisting of: project name and serial number. (Brouwer col 9, line 31 – 45, "The Systest Workbench also tracks tests within a project and projects themselves. In that regard, the following pieces of

information comprise the data record for a test instance: i) project name--entered at the system control interface and registered by the scripts in their start-up routines; ii) test name--also entered at the system control interface and duplicated in the scripts during the start-up; iii) test instance--number assigned in ascending order by the system control application each time a test is started; and iv) date and time--of test start and finish. Such a record is used as a key in the database to associate test output data (logs, metrics, alarms) with particular tests instance runs. Note that the project and test name may be entered at either the control and script areas for more flexibility. The system control data always takes precedence unless the field is left blank.")

It would have been obvious for one of ordinary skill in the art at the time of invention to incorporate the teaching of Brouwer into that of Wygodny in order to include project name and serial number into the data. Recording serial number and project name would help classifying the origin of the data collected, thus reduces the risk of data mix up.

As per claim 18, Wygodny did not disclose:

- The article of manufacture of claim 16, wherein the separator includes data selected from the group consisting of: project name and serial number.

However, Brouwer teaches:

- wherein the separator includes data selected from the group consisting of: project name and serial number. (Brouwer col 9, line 31 – 45, "The Systest Workbench also tracks tests within a project and projects themselves. In that regard, the following pieces of information comprise the data record for a test instance: i) project name--entered at the system control interface and registered by the scripts in their start-up routines; ii) test name--also entered at the system control interface and duplicated in the scripts during the start-up; iii) test instance--number assigned in ascending order by the system control application each time a test is started; and iv) date and time--of test start and finish. Such a record is used as a key in the database to associate test output data (logs, metrics, alarms) with particular tests instance runs. Note that the project and test name may be entered at either the control and script areas for more flexibility. The system control data always takes precedence unless the field is left blank.")

It would have been obvious for one of ordinary skill in the art at the time of invention to incorporate the teaching of Brouwer into that of Wygodny in order to include project name and serial number into the data. Recording serial number and project name would help classifying the origin of the data collected, thus reduces the risk of data mix up.

9. Claims **8, 19, 21, 22 and 25** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Wygodny**, in view of **Culter (USPAT 7188331, hereinafter Culter)**.

As per claim 8, Wygodny did not disclose:

- The method of claim 4, wherein the first computer system is operable in accordance with the Extensible Firmware Interface (EFI) framework specification.

However, Culter teaches:

- wherein the first computer system is operable in accordance with the Extensible Firmware Interface (EFI) framework specification. (Culter "In this example implementation, system firmware 207 comprises a Processor Abstraction Layer (PAL) 204, System Abstraction Layer (SAL) 205, and Extended Firmware Interface (EFI) 206.")

It would have been obvious for one of ordinary skill in the art at the time of invention to incorporate the teaching of Culter into that of Wygodny in order to use EFI framework. Wygodny teaches that the client program that's been monitored can be a firmware, using EFI would ensure the firmware become hardware independent, thus allow the

firmware or the client program as in Wygodny to be deployed into many different hardware environments.

As per claim 19, Wygodny did not disclose:

- The article of manufacture of claim 15, wherein the first computer system is operable in accordance with the Extensible Firmware Interface (EFI) framework specification.

However, Culter teaches:

- wherein the first computer system is operable in accordance with the Extensible Firmware Interface (EFI) framework specification. (Culter "In this example implementation, system firmware 207 comprises a Processor Abstraction Layer (PAL) 204, System Abstraction Layer (SAL) 205, and Extended Firmware Interface (EFI) 206.")

It would have been obvious for one of ordinary skill in the art at the time of invention to incorporate the teaching of Culter into that of Wygodny in order to use EFI framework. Wygodny teaches that the client program that's been monitored can be a firmware, using EFI would ensure the firmware become hardware independent, thus allow the firmware or the client program as in Wygodny to be deployed into many different hardware environments.

As per claim 21, Wygodny did not disclose:

- The computer system of claim 20, wherein the first computer system is operable in accordance with the Extensible Firmware Interface (EFI) framework specification.

However, Culter teaches:

- wherein the first computer system is operable in accordance with the Extensible Firmware Interface (EFI) framework specification. (Culter "In this example implementation, system firmware 207 comprises a Processor Abstraction Layer (PAL) 204, System Abstraction Layer (SAL) 205, and Extended Firmware Interface (EFI) 206.")

It would have been obvious for one of ordinary skill in the art at the time of invention to incorporate the teaching of Culter into that of Wygodny in order to use EFI framework. Wygodny teaches that the client program that's been monitored can be a firmware, using EFI would ensure the firmware become hardware independent, thus allow the firmware or the client program as in Wygodny to be deployed into many different hardware environments.

As per claim 22, Wygodny and Culter further teach:

- The computer system of claim 21, the debug information node includes data selected from the group of: priority, time stamp, host ID, metadata, separator,

debug information, module name, sub-module name, priority, file name, line number, project name, and serial number. (Wygodny col 8, line 21 – 33, "The debug information 121 is preferably created by a compiler when the client is compiled. Using the debug information 121 the analyzer translates function names and source lines to addresses when creating the TCI file 120. Conversely, the analyzer 106 uses the debug information 121 to translate addresses in the trace data back into function names and source lines when formatting a display for the user interface. One skilled in the art will recognize that other build information may be used as well, including, for example, information in a linker map file and the Type Library information available in a Microsoft OLE-compliant executable.')

As per claim 25, Wygodny and Culter further teach:

- The computer system of claim 21, further comprising a user interface to set the configuration parameters. (Wygodny col 2, line 51 – col 3, line11, "The present invention overcomes these and other problems associated with debugging and tracing the execution of computer programs. One aspect of the present invention is a software system that facilitates the process of identifying and

isolating bugs within a client program by allowing a developer to trace the execution paths of the client. The tracing can be performed without requiring modifications to the executable or source code files of the client program. Preferably, the trace data collected during the tracing operation is collected according to instructions in a trace control dataset, which is preferably stored in a Trace Control Information (TCI) file. Typically, the developer generates the TCI file by using a trace options editor program having a graphical user interface. The options editor displays the client's source code representation on a display screen together with controls that allow the software developer to interactively specify the source code and data elements to be traced. The options editor may use information created by a compiler or linker, such as debug information, in order to provide more information about the client and thereby make the process of selecting trace options easier. Once the trace options are selected, the client is run on a computer, and a tracing library is used to attach to the memory image of the client (the client process). The tracing library is configured to monitor execution of the client, and to collect trace data, based on selections in the trace options. The trace data

collected by the tracing library is written to an encoded buffer in memory. The data in the buffer may optionally be saved to a trace log file for later use.")

10. Claim **9** is rejected under 35 U.S.C. 103(a) as being unpatentable over **Wygodny**, in view of **Culter**, further in view of **Kaneko et al (US-PGPUB 20040199671, hereinafter Kaneko)**.

As per claim 9, Wygodny and Culter did not teach:

- The method of claim 8, further comprising: buffering the debug information node into a non-volatile memory upon failure to transmit the debug information node from the first computer system to the second computer system; and re-attempting to transfer the debug information from the buffer to the second computer system.

However, Kaneko teaches:

- buffering the debug information node into a non-volatile memory upon failure to transmit the debug information node from the first computer system to the second computer system; and re-attempting to transfer the debug information from the buffer to the second computer system. (Kaneko [0035], "Since the data is retained in the memory 3 even when the data transfer has been started but not completed due to an

obstacle or the like, the communication assisting apparatus 100 can transmit again the data from the memory in a state independent of the processing operation of the PC 111." And [0061], "Upon receiving data from a data processing apparatus of the side of transmitter (or the side of transfer originator), the communication assisting apparatus temporarily stores the data in the memory provided on the inside thereof. By this operation, the data processing apparatus located on the side of transmitter can shift to the next processing even when the data is not actually transferred. Therefore, the operating efficiency of the data processing apparatus located on the side of transmitter can be remarkably improved. In particular, the communication assisting apparatus transmits data by wireless, and therefore, the communication assisting apparatus can retransmit the data stored in the memory even if there is communication failure due to an obstacle or the like, which does not occur by wire, or random entry of an unrelated signal such as noise or the like. This can prevents from putting or reducing processing load on the data processing apparatus located on the side of transmitter. Moreover, the data processing apparatus is connected with no apparatus other than the communication

assisting apparatus with regard to the transmission of the data, and therefore, the data processing apparatus can operate without being influenced by the state of communication with the data processing apparatus located on the transfer destination side and so on. Therefore, stabilization of operation can be also achieved.")

It would have been obvious for one of ordinary skill in the art at the time of invention to incorporate the teaching of Kaneko into that of Wygodny and Culter in order buffer the node when network transmission failed and re-attempt it later, it is widely known in the field that data are often stored into buffer or memory during transmission to ensure that in the event of communication failure the data would still be available to transmit again later. Applicant have only claimed commonly known steps in the art to achieve predictable results.

11. Claim **10** is rejected under 35 U.S.C. 103(a) as being unpatentable over **Wygodny**, in view of **Culter**, further in view of **House et al (USPAT 6061517, hereinafter House)**.

As per claim 10, Wygodny and Culter further teach:

- The method of claim 8, further comprising: and receiving the debug information node from the first computer system. (Wygodny col 6, line 11 – 12, "The user 110 sends the trace log file 122 back to the developer 112."

And col 3, line 26 – 44, "In one embodiment, the software system provides a remote mode that enables the client program to be traced at a remote site, such as by the customer at a remote customer site, and then analyzed at the developer site. When the remote mode is used, the developer sends the TCI file for the particular client to a remote user site together with a small executable file called the tracing "agent." The agent is adapted to be used at the remote user site as a stand-alone tracing component that enables a remote customer, who does not have access to the source code of the client, to generate a trace file that represents execution of the client application at the remote site. The trace file is then sent to the developer site (such as by email), and is analyzed by the software developer using the analyzer. The remote mode thus enables the software developer to analyze how the client program is operating at the remote site, without the need to visit the remote site, and without exposing to the customer the source code or other confidential details of the client program.")

Wygodny and Culter did not teach:

- monitoring at the second computer system traffic of a network for a debug information node from a second computer system;

However, House teaches:

- monitoring at the second computer system traffic of a network for a debug information node from a second computer system; (House figure 1, Note the database server monitors the network server and client.)

It would have been obvious for one of ordinary skill in the art at the time of invention to incorporate the teaching of House into that of Wygodny and Culter in order to have another system monitor the network traffic, This ensures the network integrity and notify the developer about incoming report should the developer side is experiencing network issues and unable tor receive the data.

12. Claim **12** is rejected under 35 U.S.C. 103(a) as being unpatentable over **Wygodny**, in view of **Kaneko**.

As per claim 12, Wygodny did not disclose:

- The method of claim 11, further comprising: buffering the node into a storage device upon failure to transmit the node through the network adaptor.

However, Kaneko teaches:

- buffering the node into a storage device upon failure to transmit the node through the network adaptor. (Kaneko [0035], "Since the data is retained in

the memory 3 even when the data transfer has been started but not completed due to an obstacle or the like, the communication assisting apparatus 100 can transmit again the data from the memory in a state independent of the processing operation of the PC 111." And [0061], "Upon receiving data from a data processing apparatus of the side of transmitter (or the side of transfer originator), the communication assisting apparatus temporarily stores the data in the memory provided on the inside thereof. By this operation, the data processing apparatus located on the side of transmitter can shift to the next processing even when the data is not actually transferred. Therefore, the operating efficiency of the data processing apparatus located on the side of transmitter can be remarkably improved. In particular, the communication assisting apparatus transmits data by wireless, and therefore, the communication assisting apparatus can retransmit the data stored in the memory even if there is communication failure due to an obstacle or the like, which does not occur by wire, or random entry of an unrelated signal such as noise or the like. This can prevents from putting or reducing processing load on the data processing apparatus located on the side of transmitter. Moreover, the data processing

apparatus is connected with no apparatus other than the communication assisting apparatus with regard to the transmission of the data, and therefore, the data processing apparatus can operate without being influenced by the state of communication with the data processing apparatus located on the transfer destination side and so on. Therefore, stabilization of operation can be also achieved.")

It would have been obvious for one of ordinary skill in the art at the time of invention to incorporate the teaching of Kaneko into that of Wygodny in order buffer the node when network transmission failed and re-attempt it later, it is widely known in the field that data are often stored into buffer or memory during transmission to ensure that in the event of communication failure the data would still be available to transmit again later. Applicant have only claimed commonly known steps in the art to achieve predictable results.

13. Claims **23 and 24** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Wygodny**, in view of **Culter**, further in view of **Rothman et al (US-PGPUB 20050240828, hereinafter Rothman)**.

As per claim 23, Wygodny and Culter did not teach:

- The computer system of claim 21, wherein the network adaptor is a wired Ethernet card.

However, Rothman teaches:

- wherein the network adaptor is a wired Ethernet card. (Rothman [0045], ". The example processor system 600 may also include an adapter card 630 operatively coupled to a display device 632 and a network adapter 636 such as, for example, an Ethernet card or any other card that may be wired or wireless.")

It would have been obvious for one of ordinary skill in the art at the time of invention to include the teaching of Rothman into that of Wygodny and Culter in order to . Wygodny discloses that the debug information from a client can be sent to the developer for debugging through the internet (Wygodny col 3, line 26 – 44,), it would have be obvious to send those data from wired or wireless Ethernet card to ensure fast data transmission. Applicant have only claimed known methods in the art to achieve predictable goal.

As per claim 24, Wygodny and Culter did not teach:

- The computer system of claim 21, wherein the network adaptor is a wireless Ethernet card.

However, Rothman teaches:

- wherein the network adaptor is a wireless Ethernet card. (Rothman [0045], ". The example processor system 600 may also include an adapter card 630 operatively coupled to a display device 632 and a network adapter 636 such as, for example, an Ethernet card or any other card that may be wired or wireless.")

It would have been obvious for one of ordinary skill in the art at the time of invention to include the teaching of Rothman into that of Wygodny and Culter in order to . Wygodny discloses that the debug information from a client can be sent to the developer for debugging through the internet (Wygodny col 3, line 26 – 44,), it would have be obvious to send those data from wired or wireless Ethernet card to ensure fast data transmission. Applicant have only claimed known methods in the art to achieve predictable goal.

Conclusion

14. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Charles Swift whose telephone number is (571)270-7756. The examiner can normally be reached on Monday through Thursday, 9:00AM to 6:00PM, Friday 10:30AM - 3:30PM, Eastern Time.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Emerson Puente can be reached on (571)272-3652. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Charles Swift/
Examiner, Art Unit 2191

/Emerson C Puente/
Supervisory Patent Examiner, Art Unit 2195